L3: Entry 2 of 410 File: PGPB Jul 26, 2001

DOCUMENT-IDENTIFIER: US 20010010090 A1
TITLE: Method for design optimization using logical and physical information

BSTX:
[0019] The method of the present invention can be implemented in a parallel processing design automation system to achieve high performance. Such a parallel processing system includes (a) multiple central processing units (CPUS) executing independently of each other; (b) a shared memory accessible by each CPU for holding data structures of the integrated circuit design; and (c) a control program managing a task list which specifies tasks for optimizing the integrated design. The control program assigns tasks on the task list to each of the CPUs. The tasks include placement, performance analysis, logic optimization and routing. The control program provides a locking mechanism for locking data structures stored in the shared memory to support concurrent operations by the CPUs each accessing the integrated circuit design represented by the data structures. Further, the parallel processing design automation system supports multithread execution. Parallelism is further enhanced by adopting algorithms for placement, timing analysis, logic optimization and routing suitable for parallel execution by the multiple CPUs each accessing the design data structures stored in the shared memory.

DETX:
[0048] FIG. 4 shows a parallel processing system 400 suitable for implementing a concurrent design optimizer of the present invention. As shown in FIG. 4, parallel processing computer system 400 includes a shared memory 420 and a large number of central processing units (CPUs) 410-1, 410-2, 410-3, . . . 410-(m-1) to 410-m. Provided in shared memory 420 is an operating system 409 which can be executed on each of CPUs 410-1 to 410-m. A thread library 408 provides the control programs for executing multiple concurrently executed threads on operating system 409. The concurrent design optimizer provides data structure 401, representing the design to be optimized, in shared memory 420. Also residing in shared memory 420 are (a) interface and initialization module 403, representing the programs for interface with the design database and a user interface, (b) placement module 404, representing the programs for placement of clusters and for placement optimization; (c) timing/power analysis module 405, representing programs for incremental timing and power analysis steps, (d) logic optimization module 406, representing programs for optimizing a logic circuit, and (e) routing or routing estimation module 407, representing programs for routing and estimating use of routing resources. Alternatively, each CPU can have its own copy of modules 403-407, residing either in shared memory 420 or in a memory to which the CPU has exclusive access (e.g., a private address space).

L3: Entry 6 of 410 File: USPT Aug 14, 2001

DOCUMENT-IDENTIFIER: US 6275980 B1
TITLE: Programming method for concurrent programs and program supporting apparatus thereof

BSPR:
Referring to FIG. 1A, a process P1 performs initialization (init) of a shared memory M, a process P2 performs read access (read) to the shared memory M, and a process P3 performs write access (write) to the shared memory M. When these processes are operated in in concurrent processing system which executes these processes by different processors, a total of six (FIG. 1B) combinations of operations are available. Normally, the system starts processing with initialization. Assume that a correct result is obtained when the program is operated in an order of processes P1 (init).fwdarw.P2 (read).fwdarw.P3 (write) or P1 (nit).fwdarw.P3 (write).fwdarw.P2 (read). In this case, as for the four remaining combinations (e.g., P2 (read).fwdarw.P3 (write).fwdarw.P1 (init)), since initialization is not performed first, a correct result cannot be obtained.

DEPR:
FIG. 2 is a block diagram showing the arrangement of a computer system for realizing a supporting apparatus for programming a concurrent program according to the present invention. Referring to FIG. 2, N processors 1-1, 1-2, . . . , 1-N can simultaneously execute a concurrent program and access a shared memory 3 and peripheral devices through an I/O interface 2. The peripheral devices are constituted by an input unit 4, an output unit 5, and an external storing unit 6.

DEPR:
The concurrent program in FIG. 5 is constituted by a process P1 and a process P2. These processes are realized as entities only when an execute module generated upon compiling source codes of the concurrent program is executed in a computer. Concurrent programs respectively corresponding to the processes P1 and P2 need not be stored in storing media physically independent of each other. A shared memory M here represents an external memory. The shared memory M can perform write/read access in accordance with an access instruction of the concurrent program. Solid arrows in FIG. 5 represent access to the shared memory M upon execution of the processes P1 and P2.

DEPR:
The second embodiment has the following target concurrent program as in the first embodiment. The concurrent program is constituted by a plurality of processes. The concurrent program is executed by a multiprocessor of a shared memory type. A processor (CPU) is assigned to each process. Synchronization among processes is realized by a basic synchronization instruction and the shared memory.

DEPR:
FIG. 14 is a view showing a concurrent program. Processes P1 and P2 are operated in a concurrent manner. The processes P1 and P2 access a shared memory M.